Laboratoire Greyc
Université de Caen Normandie

Optimalité et analyse en moyenne de l'algorithme Git bisect

Julien Courtiel (julien.courtiel@unicaen.fr),
Paul Dorbec (paul.dorbec@unicaen.fr)

Normandie Univ, UNICAEN, ENSICAEN, CNRS,
GREYC, 14000 Caen, France

Au cours de ce stage, nous analyserons un algorithme issu du logiciel de gestion de versions *git*, communément appelé `git bisect`. Cet algorithme explore l'ensemble des révisions (*commits*) d'un dépôt pour rechercher la révision par laquelle un bug a été introduit.

Concrètement, l'ensemble des versions d'un projet peut-être modélisé par un graphe orienté dont les sommets sont les versions, et les arêtes les révisions. On notera que ce graphe est particulier. D'une part, il ne comporte pas de cycle (et appartient donc à la famille des *DAG*, pour *Directed Acyclic Graph*). D'autre part, chaque sommet a au plus deux arêtes entrantes, puisque les fusions n'impliquent que deux branches. Le degré sortant, lui, n'est pas borné.

L'algorithme `git bisect` que l'on souhaite étudier applique des tests aux différentes versions, tests pouvant potentiellement être lents. Chaque test indique si oui ou non le bug est déjà présent dans la version correspondante. Un algorithme naïf qui testerait toutes les versions trouverait à coup sûr la révision d'introduction du bug, mais serait très coûteux en temps. Dans `git bisect`, seul un sous-ensemble des versions est testé, en utilisant une heuristique inspirée de la recherche dichotomique.

Le problème d'optimisation correspondant au plus petit nombre de tests à effectuer pour trouver la révision d'introduction du bug est réputé NP-complet. Néanmoins, l'algorithme `git bisect` semble expérimentalement performant. Le but de ce stage est de vérifier cette hypothèse et de comprendre pourquoi. En particulier, le stage consistera à :

- Proposer un modèle aléatoire de graphes de versions.
- Analyser l'espérance du nombre de tests effectués par `git bisect`.
- Déterminer si l'algorithme `git bisect` est optimal pour l'ensemble des graphes de version.
- S'il n'est pas optimal, vérifier son comportement sur le modèle aléatoire, et la distance des solutions fournies avec l'algorithme optimal.

Le cas de familles de graphes plus générales (par exemple de degré entrant borné par une constante supérieure) et d'autres algorithmes pourront être considérés par la suite.

Le stage se déroulera à Caen, au Greyc, dans l'équipe AmacC. S'il est concluant, il pourrait se poursuivre par une offre de thèse.